Rule-based modelling with BioNetGen

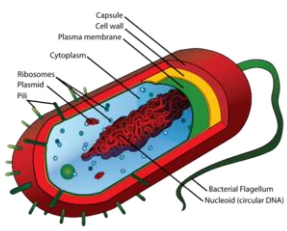
Workshop Overview

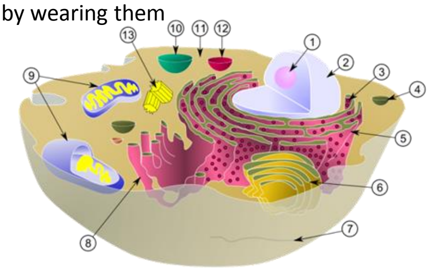
- What is modelling?
- Why rule-based modelling?
- Deterministic and Stochastic modelling
- Michaelis-Menten model
- BioNetGen practice

What is a Model?

Taken from the Oxford dictionary:

- A three-dimensional representation of a person or thing or of a proposed structure, typically on a smaller scale than the original
- A thing used as an example to follow or imitate
- A simplified description, especially a mathematical one, of a system or process, to assist calculations and predictions
- A person employed to display clothes by wearing them



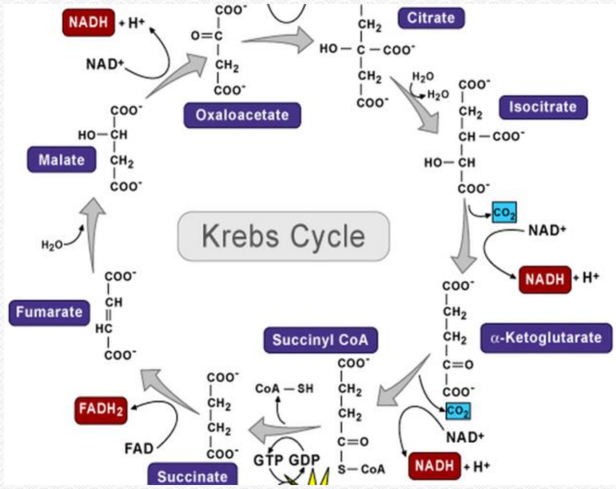


Source: http://oxforddictionaries.com/definition/english/model, http://commons.wikimedia.org

Modelling in Synthetic Biology

- Modelling traditionally used in biology to understand and imitate systems
- Synthetic Biology utilizes modelling as a design to follow
- e.g. Predicting the effect of adding a BioBrick into our bacteria

Modelling Biological Pathways



Source: http://www.npr.org/blogs/krulwich/2011/09/14/140428189/lord-save-me-from-the-krebs-cycle

Diagrammatic Model

- Useful to visualise the reactions taking place
- Can't predict what will happen if you change the reaction conditions e.g. add more of one reactant
- Can't predict how the concentration of molecules will change with time

Mathematical Model

- Predict changes in a system as time progresses
- What can change? Concentrations of…
  - Reactants
  - Products
  - Enzymes
- Which might lead to a change in the rate of reactions

Mathematical Model

- Often simulated as:
- Ordinary differential equation (ODE)
  - Deterministic
- Stochastic
  - Non-deterministic
  - Random process which evolves in time

Introduction to SBML

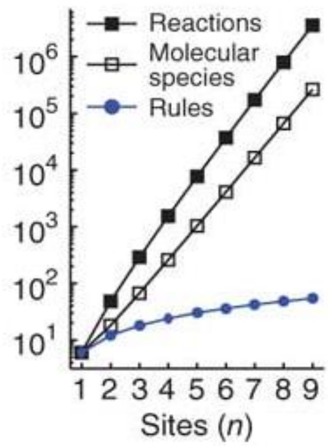
- **Systems Biology Markup Language**
- Often used to write models in synthetic biology
- Difficult to write from scratch
- A common language for communication between software packages

What is Rule-Based Modelling?

- Each molecule contains domains that can link to other molecules
- Complexes are built up by assembly of binding molecules
- Rules specify which reactions can occur and at what rate

Why Rule-Based Modelling?

- In modelling, each 'different state' is usually treated as a separate species
- But with rule-based modelling you can define a molecule with multiple states
- This saves time and effort



Sneddon MW, Faeder JR and Emonet T. Efficient modeling, simulation and coarse-graining of biological complexity with NFsim. Nature Methods (2011) 8(2):177-83.

Michaelis-Menten (MM) Rate Law

- Enzyme + Substrate ←→ Enzyme-Substrate Complex
- Enzyme-Substrate Complex → Product + Enzyme
- The M-M rate law $$v = \frac{V_{\max}[S]}{K_m + [S]}$$
- $V_{max} = k_2 (E + [ES])$
- $K_M = \frac{k_1 + k_2}{k_1}$

Michaelis-Menten Rate Law

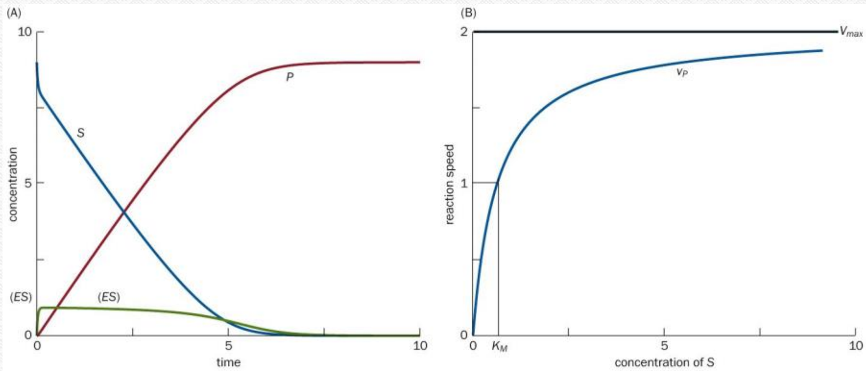


Figure 8.3 A First Course in Systems Biology (© Garland Science 2013)

Simple Michaelis–Menten Reaction


```xml
<?xml version="1.0" encoding="UTF-8"?>
<!-- Created by BioNetGen 2.2.4 -->
<sbml xmlns="http://www.sbml.org/sbml/level2" level="2" version="1">
 <model id="MM-ssa">
  <listOfCompartments>
   <compartment id="cell" size="1"/>
  </listOfCompartments>
  <listOfSpecies>
   <species id="S1" compartment="cell" initialConcentration="0" name="S(a)"/>
   <species id="S2" compartment="cell" initialConcentration="11999" name="E(a)"/>
   <species id="S3" compartment="cell" initialConcentration="30099" name="P()"/>
   <species id="S4" compartment="cell" initialConcentration="1" name="E(a!1).S(a!1)"/>
  </listOfSpecies>
  <listOfParameters>
   <!-- Independent variables -->
   <parameter id="c1" value="0.00166"/>
   <parameter id="c1r" value="0.0001"/>
   <parameter id="c2" value="0.1"/>
   <!-- Dependent variables -->
   <!-- Observables -->
   <parameter id="Enzyme" constant="false"/>
   <parameter id="Substrate" constant="false"/>
   <parameter id="Product" constant="false"/>
   <parameter id="Complex" constant="false"/>
   <!-- Global functions -->
  </listOfParameters>
  <listOfRules>
   <!-- Dependent variables -->
   <!-- Observables -->
   <assignmentRule variable="Enzyme">
     <math xmlns="http://www.w3.org/1998/Math/MathML">
      <apply>
       <plus/>
       <cn> 0 </cn>
       <ci> S2 </ci>
      </apply>
     </math>
   </assignmentRule>
   <assignmentRule variable="Substrate">
     <math xmlns="http://www.w3.org/1998/Math/MathML">
      <apply>
       <plus/>
       <cn> 0 </cn>
       <ci> S1 </ci>
      </apply>
     </math>
   </assignmentRule>
   <assignmentRule variable="Product">
     <math xmlns="http://www.w3.org/1998/Math/MathML">
      <apply>
       <plus/>
       <cn> 0 </cn>
       <ci> S3 </ci>
      </apply>
     </math>
   </assignmentRule>
   <assignmentRule variable="Complex">
     <math xmlns="http://www.w3.org/1998/Math/MathML">
      <apply>
       <plus/>
       <cn> 0 </cn>
       <ci> S4 </ci>
      </apply>
     </math>
   </assignmentRule>
   <!-- Global functions -->
  </listOfRules>
  <listOfReactions>
   <reaction id="R1" reversible="false">
    <listOfReactants>
     <speciesReference species="S1"/>
     <speciesReference species="S2"/>
    </listOfReactants>
    <listOfProducts>
     <speciesReference species="S4"/>
    </listOfProducts>
    <kineticLaw>
     <math xmlns="http://www.w3.org/1998/Math/MathML">
      <apply>
       <times/>
       <ci> c1 </ci>
       <ci> S1 </ci>
       <ci> S2 </ci>
      </apply>
```

Simple Michaelis–Menten Reaction

```xml
</math>
    </kineticLaw>
</reaction>
   <reaction id="R2" reversible="false">
    <listOfReactants>
     <speciesReference species="S4"/>
    </listOfReactants>
    <listOfProducts>
     <speciesReference species="S1"/>
     <speciesReference species="S2"/>
    </listOfProducts>
    <kineticLaw>
     <math xmlns="http://www.w3.org/1998/Math/MathML">
      <apply>
       <times/>
       <ci> c1r </ci>
       <ci> S4 </ci>
      </apply>
     </math>
    </kineticLaw>
   </reaction>
   <reaction id="R3" reversible="false">
    <listOfReactants>
     <speciesReference species="S4"/>
    </listOfReactants>
    <listOfProducts>
     <speciesReference species="S2"/>
     <speciesReference species="S3"/>
    </listOfProducts>
    <kineticLaw>
     <math xmlns="http://www.w3.org/1998/Math/MathML">
      <apply>
       <times/>
       <ci> c2 </ci>
       <ci> S4 </ci>
      </apply>
     </math>
    </kineticLaw>
   </reaction>
  </listOfReactions>
 </model>
</sbml>
```

- Writing models by hand in SBML is clearly difficult
  - Written for the computer, not the user
  - Every potential interaction between species must be contained within the model

Example of a Michaelis-Menten ODE

```
# A simple Michaelis-Menten enzymatic reaction
begin parameters
    k1      1e6
    k1r     1e-4
    k2      0.1
end parameters

begin molecule types
    S(a)
    E(a)
    P()
end molecule types
begin seed species
    S(a)        5e-5
    E(a)        2e-5
    P()         0
end seed species
# the actual reactions
begin reaction rules
    # Binding and dissociation
    S(a) + E(a) <-> S(a!1).E(a!1)    k1, k1r
    # Production
    S(a!1).E(a!1) -> P() + E(a) k2
end reaction rules
begin observables
    Molecules   Enzyme      E(a)
    Molecules   Substrate   S(a)
    Molecules   Product     P()
    Molecules   Complex     S(a!1).E(a!1)
end observables
## actions ##
generate_network({overwrite=>1})
# Equilibration
simulate_ode({t_end=>100,n_steps=>1000,atol=>1e-10,rtol=>1e-8,sparse=>1})
```

Parameters which define the rate of reaction

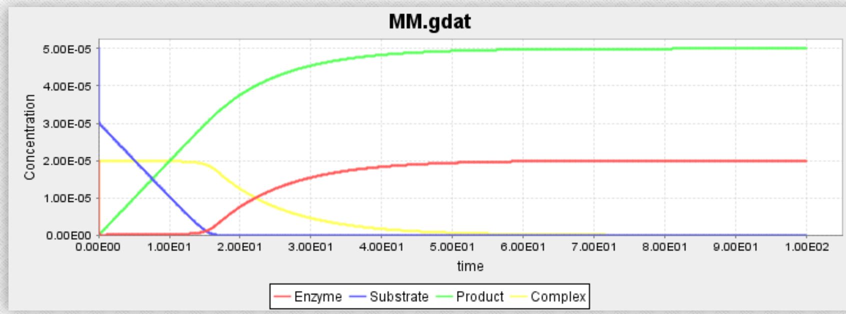
List of the molecules involved in the reaction

Starting concentrations of molecules

Reactions that are going to be simulated

Declaration of names of species which will be measured in the graph

Michaelis-Menten Simulation



Stochastic Modelling

- ODE only accurate at high concentrations
- Otherwise use stochastic modelling
- Quantities expressed as **numbers** of molecules
- Reactions only occur with certain probability
  - Because reactions occur when molecules collide randomly
- Uses stochastic simulation algorithm (SSA)

Stochastic Modelling

- We now deal with integer quantities (of chemical species) instead of continuous quantities
- $X_1 + X_2 \rightarrow X_3$ (one molecule of X_1 plus one molecule of X_2 are transformed into one molecule of X_3)
- Let's simulate one step

$$\begin{array}{lcl} X_1 = 5 & & X_1 = 4 \\ X_2 = 5 & \longrightarrow & X_2 = 4 \\ X_3 = 0 & & X_3 = 1 \end{array}$$

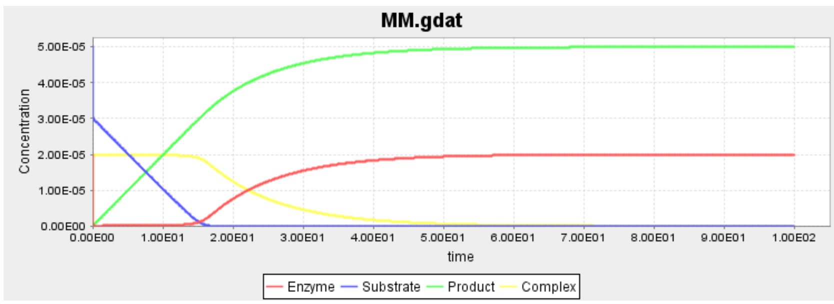
- Note: You should avoid using more than 2 reagents in a reaction e.g. $X_1 + X_2 + X_3 \rightarrow X_4$

ODE vs SSA

When the number of molecules is high SSA behaves like ODE(!)

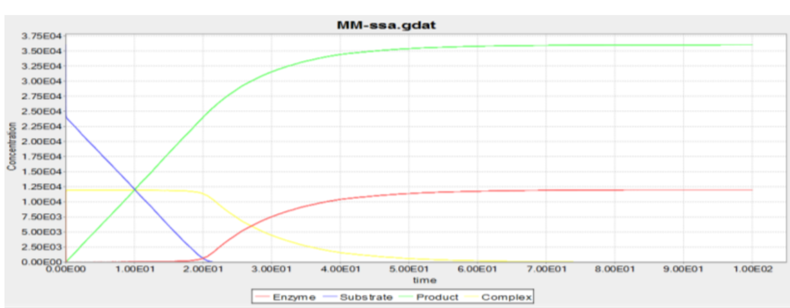
ODE vs SSA

- Deterministic simulation of MM at high concentrations (E = $2*10^{-5}$, S = $5*10^{-5}$)



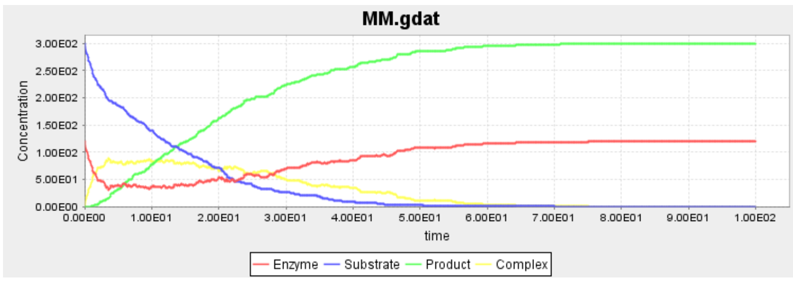
ODE vs SSA

- Stochastic simulation of MM at high molecule numbers (start with E = 12 000, S = 36 000)



ODE vs SSA

- Stochastic simulation of MM at low molecule numbers (E =100, S = 300)

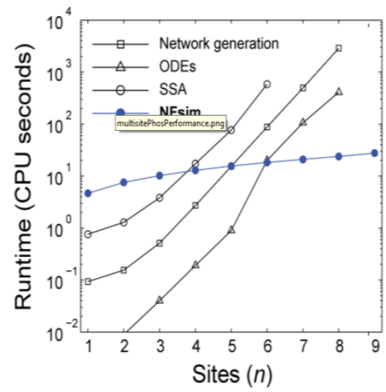


Which Type of Model to Use?

- At **low** molecule numbers
  - use Stochastic model
- At **high** molecule numbers
  - Both types of simulation can be used
  - But SSA might require more “computing power” than the ODE simulation

NFsim

- Complex pathways are time consuming to simulate – have to keep track of every state/interaction
- NFsim, a BioNetGen add-on, runs network free simulations- only tracks the existing system
- Much lower runtime



Runtime increases exponentially with complexity, except for NFsim

Diagram taken from NFsim user manual

Examples

Before going any further into examples, do You have any questions?

Fundamental Operations BioNetGen

Script is split into 'blocks', each defining a different part of the model

- **Parameters:** reaction rate constants, and values for initial concentrations of species in the biological system
- **Molecule types**: The molecules the model contains, including their components and allowed component states (e.g. phosphorylation sites)
- **Seed species:** The initial state of system (initial species and their concentrations)
- **Observables:** The model outputs

Fundamental Operations BioNetGen

Script is split into 'blocks', each defining a different part of the model

- **Functions:** Define global and/or local functions of observables for use in rate laws. Not essential.
- **Reaction rules:** Rules that describe how molecules interact
- **Actions:** Network generation and simulation

Each block must be enclosed with 'begin *x*' and 'close *x*' where '*x*' is the block in question (e.g. 'begin parameters' and 'end parameters')

Fundamental Operations BioNetGen

- To define a molecule ‘a’ with 2 binding sites:
  - a(b,c)
  - Where the names of the binding sites are unimportant
- To define a phosphorylation site ‘ps’ on molecule ‘a’
  - a(ps~U~P)
  - Where U represents one state (unphosphorylated) and P represents the other state (phosphorylated)

Fundamental Operations BioNetGen

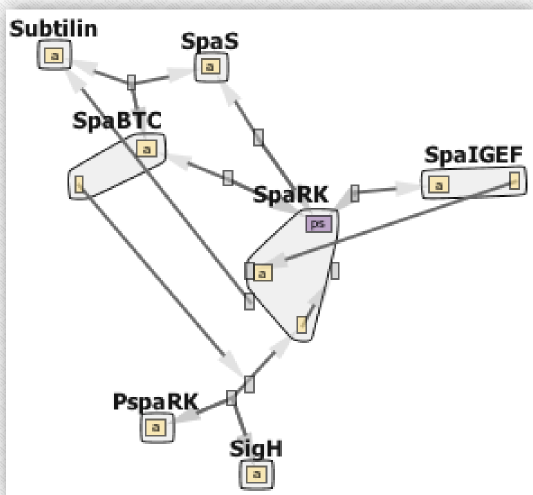
- To bind molecules ‘a’ and ‘b’, with binding sites (b) and (a):
  - a(b) + b(a) -> a(b!1).b(a!1)
- For the phosphorylation at site ‘ps’ on molecule ‘a’
  - a(ps~U) -> a(ps~P)
  - Where molecule a has previously been defined as a(ps~U~P)

Thank you for your attention ☺

Subtilin production

- Natural antibiotic secreted by *B.subtilis* in response to excessive growth
- Lack of food activates:
  - Subtilin production
  - Immunity of cells
- This system can be modeled using BioNetGen

Subtilin system



33

Modelling

- Not possible to model a population in BioNetGen
- But there is a way to get around it!
- A set of rules to mimic the effects of the changes in population can be created
- Set of logical functions is introduced

Example...

34

Single Cell vs Hybrid Models

Subtilin synthesis model.bngl

```
# Two-component subtilin synthesis single cell model
begin parameters
    k1  1e6
    k1r 1e-4
    kp1 1e6
    k3  5e4
    k4  1e4
    k5  0.25e4
    k6  1e4
    kdp1 1e1
        kdeg 0.1
        kdeg1 0.1
end parameters


begin molecule types
    SigH(a)
    PspaRK(a)
    SpaRK(a,ps~U~P)
    SpaBTC(a)
    SpaIGEF(a)
    SpaS(a)
    Subtilin(a)
end molecule types

begin seed species
    SigH(a)         2e-6
    PspaRK(a)       2e-10
    Subtilin(a)     2e-7
end seed species


# the actual reactions
begin reaction rules
    # Synthesis of Spa RK
    SigH(a) + PspaRK(a) -> SpaRK(a,ps~U) + SigH(a) + PspaRK(a)  k1
    # Detection of external Subtilin
    Subtilin(a) + SpaRK(a,ps~U) -> SpaRK(a,ps~P) + Subtilin(a) kp1
    # Activation of Spa
    SpaRK(ps~P) -> SpaS(a) + SpaRK(ps~P)    k3
    SpaRK(ps~P) -> SpaBTC(a) + SpaRK(ps~P)  k4
    SpaRK(ps~P) -> SpaIGEF(a) + SpaRK(ps~P) k5
    SpaRK(ps~P)-> SpaRK(ps~U) kdp1
    # Conversion into subtilin
```

test.bngl

```
 A model of subtilin biosynthesis (hybrid)
egin parameters
   k1  1e6
   k1r 1e-4
   kp1 1e6
   k3  5e4
   k4  1e4
   k5  0.25e4
   k6  1e3
   kdp1 12
   kdeg 0.00001
   kdeg1 0.001
nd parameters



egin molecule types
   SigH(a)
   PspaRK(a)
   SpaRK(a,ps~U~P)
   SpaBTC(a)
   SpaIGEF(a)
   SpaS(a)
   Subtilin(a)
   Food()
nd molecule types

egin seed species
   SigH(a)         2e-6
   PspaRK(a)       2e-10
   Subtilin(a)     2e-7
   Food()          1e-4
nd seed species

egin observables
   Molecules   SigH SigH(a)
   Molecules   Promoter PspaRK(a)
   Molecules   SpaRK SpaRK(a,ps~U)
   Molecules   SpaRK1 SpaRK(a,ps~P)
   Molecules   SpaBTC SpaBTC(a)
   Molecules   SpaIGEF SpaIGEF(a)
   Molecules   SpaS SpaS(a)
   Molecules   Subtilin Subtilin(a)
   Molecules   f_obs Food()
```

Single Cell vs Hybrid Models

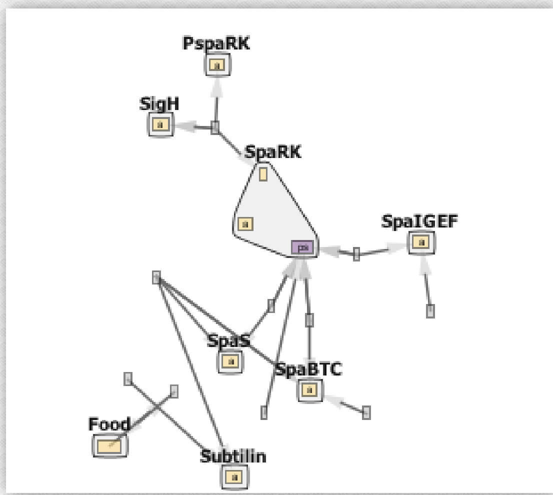
Subtilin synthesis model.bngl

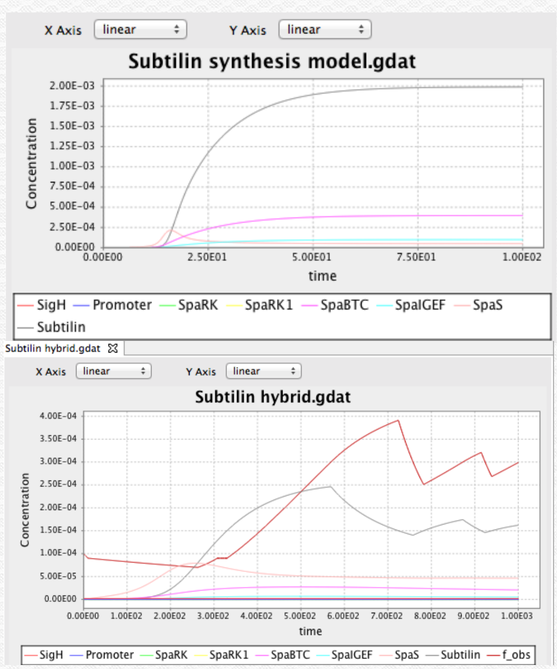
```

begin seed species
    SigH(a)         2e-6
    PspaRK(a)       2e-10
    Subtilin(a)     2e-7
end seed species


# the actual reactions
begin reaction rules
    # Synthesis of Spa RK
    SigH(a) + PspaRK(a) -> SpaRK(a,ps~U) + SigH(a) + PspaRK(a)  k1
    # Detection of external Subtilin
    Subtilin(a) + SpaRK(a,ps~U) -> SpaRK(a,ps~P) + Subtilin(a) kp1
    # Activation of Spa
    SpaRK(ps~P) -> SpaS(a) + SpaRK(ps~P)    k3
    SpaRK(ps~P) -> SpaBTC(a) + SpaRK(ps~P)  k4
    SpaRK(ps~P) -> SpaIGEF(a) + SpaRK(ps~P) k5
    SpaRK(ps~P)-> SpaRK(ps~U) kdp1
    # Conversion into subtilin
    SpaS(a) + SpaBTC(a) -> Subtilin(a) + SpaBTC(a) k6
    SpaRK() -> 0 kdeg1
    SpaBTC() -> 0 kdeg1
    SpaIGEF() -> 0 kdeg1
        Subtilin(a) -> 0 kdeg
end reaction rules


begin observables
    Molecules   SigH SigH(a)
    Molecules   Promoter PspaRK(a)
    Molecules   SpaRK SpaRK(a,ps~U)
    Molecules   SpaRK1 SpaRK(a,ps~P)
    Molecules   SpaBTC SpaBTC(a)
    Molecules   SpaIGEF SpaIGEF(a)
    Molecules   SpaS SpaS(a)
    Molecules   Subtilin Subtilin(a)
end observables

## actions ##
generate_network({overwrite=>1})

# Equilibration
simulate_ode({t_end=>100,n_steps=>1000,atol=>1e-10,rtol=>1e-8,sparse=>1})
```

Subtilin hybrid.gdat | *Subtilin hybrid.bngl

```
#Functions which relate single cell model to the change in the population

begin functions
    Spark_trnscr() if(f_obs > 1e-4, 0, k1)

    f_decr1() if(Subtilin < 1.5e-4 && f_obs > 9e-5, 0.01, 0.001)

    f_incr() if(Subtilin > 0.8e-4, Subtilin*0.005, 0)
    Subt_decr() if(f_obs > 3e-4, 0.01, 0.005)
end functions

# the actual reactions
begin reaction rules
    # Synthesis of Spa RK
    SigH(a) + PspaRK(a) -> SpaRK(a,ps~U) + SigH(a)  Spark_trnscr()
    # Detection of external Subtilin
    Subtilin(a) + SpaRK(a,ps~U) -> SpaRK(a,ps~P) + Subtilin(a) kp1
    # Activation of SpaRK
    SpaRK(ps~P) -> SpaS(a) + SpaRK(ps~P)    k3
    SpaRK(ps~P) -> SpaBTC(a) + SpaRK(ps~P)  k4
    SpaRK(ps~P) -> SpaIGEF(a) + SpaRK(ps~P) k5
    SpaRK(ps~P) -> SpaRK(ps~U) kdp1

    #Degradation of made products
    SpaRK(ps~U) -> 0 0.01
    SpaBTC(a) -> 0 0.01
    SpaIGEF(a) -> 0 0.01

    #Decrease in subtilin as production stops and it diffuses away/degraded by IGEF
    Subtilin(a) -> 0 Subt_decr()

    # Conversion of SpaS into subtilin
    SpaS(a) + SpaBTC(a) -> Subtilin(a) + SpaBTC(a) k6

    #Change in food do to population increase/decrease
    0 -> Food()     f_incr()#p_decr()
    Food() -> 0     f_decr1()
end reaction rules



## actions ##
generate_network({overwrite=>1})
```

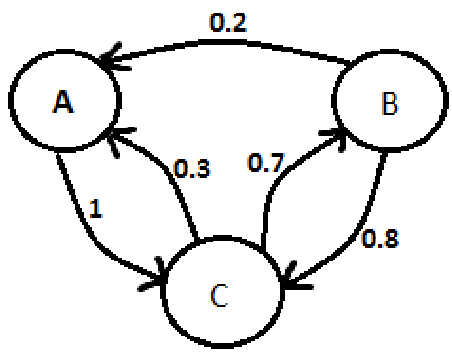
Hybrid Model





A discrete-time Markov chain

Example of Markov chain



All outgoing

Rate parameter conversion

- Most of the papers use deterministic models
- We might want to convert deterministic model (ODE) into non-deterministic (Stochastic)
  - We would need to convert deterministic rate constants (the k_j's) into stochastic rate constants (the c_j's)
- First thing to do would be to deal with concentrations:
  - Convert M (moles per litre) into molecule numbers

Concentrations and molecules example

- For example let's try to convert M of the enzyme RibA to molecules number
- $[RibA] = 5 * 10^{-7}$ M
- Volume of the cell is $V= 10^{-15}$ litres
- Number of moles in the RibA enzyme is $[RibA]V= 5 * 10^{-22}$
- To get number of molecules we need to multiply moles with Avogadro number n_A

$n_A[RibA]V = n_A * 5 * 10^{-22}$

Ordinary Differential Equation (ODE)

- It works under three assumptions

1. Reactions always in **well-stirred**, homogenous media (mass action kinetics)
2. **Quasi-steady state** assumption and **substrate >> enzyme** (Michaelis-Menten rate law)
3. Concentrations are **not small** (so we can use ODE's)

Stochastic modelling

- The state space of a stochastic model is thus a set of tuples
- E.g., the state of a 6-species model is a tuple

  (x_1,x_2,x_3,x_4,x_5)

  where each x_i is a **natural number**
- In theory, the state space can be infinite (the number of tuples, not the length of each tuple)

Stochastic simulation algorithm (SSA)

- Initially developed to analyse and better understand various chemical reactions which include large number of species
- Suppose system includes M chemical reactions { $R_1,\ldots,R_M$} and N chemical species
- $x(t) = (x_1(t),\ldots,x_n(t))$ is the state vector(number of molecules of species) of the system at a time t.
- When reaction R_j fires, the system changes as

  $x(t) \to x(t) + v_j$
- v_j is vector of N integers and represents state change caused by the firing of R_j

SSA

```
//initialize time and system state
t_sim := 0;
x := x_0;
// simulation up to time T
while t_sim <= T do
        evaluate a_j(x) (1 <= j <= M) and a_0(x);
        t := sample time step from density of Eq.1;
        j:= sample reaction index from distribution of Eq.2;
        t_sim:= t_sim + t;
        x:= x+ v_j;
end
```